

AUTO APRENDIZAJE

Excepciones en Java:

NullPointerException:

Especificación:

public class **NullPointerException** extends RuntimeException

¿Cuándo Ocurre?:

Esta excepción es lanzada por la JVM cuando se realiza alguna operación sobre un objeto o identificador **null** o se invoca algún método de ese objeto.

¿Cómo se evita?:

En este ejemplo, la excepción **NullPointerException** es lanzada:

```
1. String f = null;
2. System.out.println("Esto causará que se dispare la excepción
   NullPointerException " + f.length());
```

Esta es la forma que se puede evitar que la excepción se dispare:

```
1. String f = null;
2. if(f != null)
3. System.out.println("Esto causará que se dispare la excepción
   NullPointerException "+ f.length());
```

Ejemplo:

```
1. public class EjemploNullPointerException
2. {
3.     public static void main( String[] args )
4.     {
5.         Carro carro = null;
6.
7.         if( carro != null )
8.         {
9.             // cambia el color
10.            carro.establecerColor( "Gris" );
11.            System.out.println( carro.obtenerColor() );
12.        }
13.        else
14.        {
15.            System.out.println( "El identificador no hace
referencia a ningún objeto. Es null" );
16.        }
17.    } // fin del método main
18.} // fin de la clase Ejemplo
```

```

19.
20.class Carro
21.{
22.    private String color = "Negro";
23.    private String marca = "Chevrolet";
24.
25.    public void establecerColor( String c )
26.    {
27.        c = color;
28.    } // fin del método establecerColor
29.
30.    public String obtenerColor()
31.    {
32.        return color;
33.    } // fin del método obtenerColor
34.} // fin de la clase Carro

```

Enlace al código fuente anterior:

<http://paste.ubuntu.com/704712/>

ClassNotFoundException:

Especificación:

public class **ClassNotFoundException** extends [Exception](#)

¿Cuándo Ocurre?:

Esta excepción es lanzada cuando una aplicación trata de cargar/encontrar una clase.

¿Cómo se evita?:

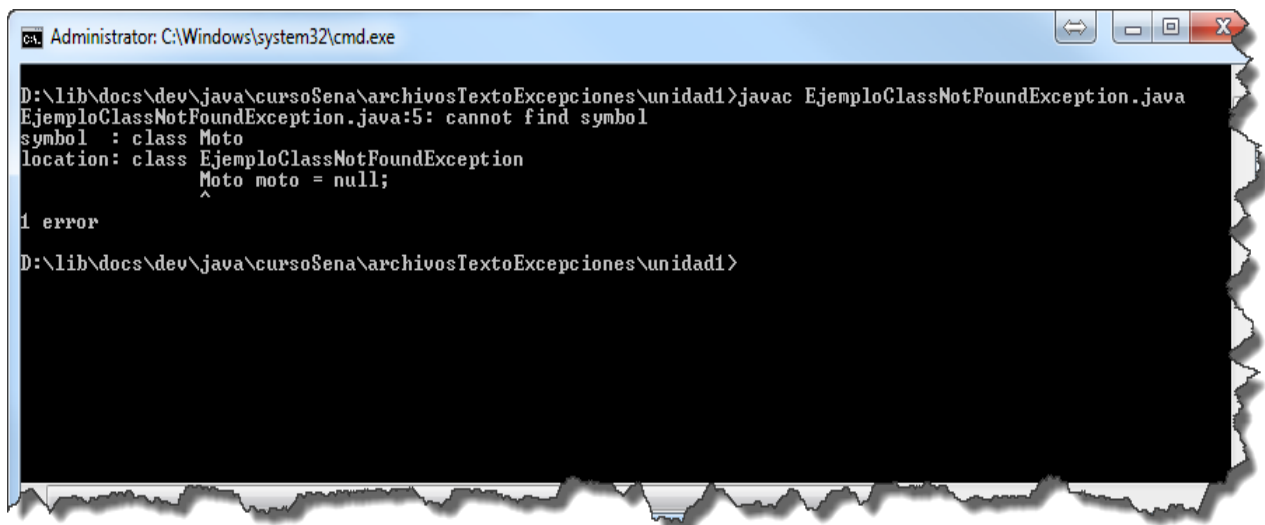
En este ejemplo, la excepción **ClassNotFoundException** es lanzada:

```

1. public class EjemploClassNotFoundException
2. {
3.     public static void main( String[] args )
4.     {
5.         Moto moto = null;
6.
7.         if( moto != null )
8.         {
9.             // cambia el color
10.            moto.establecerColor( "Gris" );
11.            System.out.println( moto.obtenerColor() );
12.        }
13.        else
14.        {
15.            System.out.println( "El identificador no hace
referencia a ningún objeto. Es null" );
16.        }
17.    } // fin del método main
18.} // fin de la clase Ejemplo

```

La salida en la línea de comandos:

A screenshot of a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The window shows the following text:

```
D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>javac EjemploClassNotPoundException.java
EjemploClassNotPoundException.java:5: cannot find symbol
symbol   : class Moto
location: class EjemploClassNotPoundException
    Moto moto = null;
    ^
1 error
D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>
```

La solución a este problema podría ser:

- Localizar la clase en el paquete correspondiente, o agregar el archivo *jar* que la contiene.
- Crear y compilar el archivo de código fuente Java.

ClassCastException:

Especificación:

public class **ClassCastException** extends RuntimeException

¿Cuándo Ocurre?:

Esta excepción es lanzada cuando por Java cuando se intenta promocionar o convertir un objeto de una tipo de dato a otro..

¿Cómo se evita?:

En este ejemplo, la excepción **ClassCastException** es lanzada:

```
import java.util.Vector;

public class EjemploClassCastException
{
    // punto de entrada de la aplicación
    public static void main(String[] args)
    {
        Vector v = new Vector();
        int i = 10;
```

```

    Object obj = v.add(i);

    try
    {
        String y = (String)obj;
        v.add(y);
    }
    catch (ClassCastException e)
    {
        System.out.println("La clase a la que pertenece: " + obj.getClass().getName()
        );
        e.printStackTrace();
    } // fin de catch
} // fin de main
} // fin de la clase EjemploCastException

```

Enlace: <http://paste.ubuntu.com/704867/>

Para evitar esta excepcion es necesario agregar `Integer y = new Integer((String) obj)` en lugar de `String y = (String) obj`:

```

    try
    {
        Integer y = new Integer((String)obj);
        v.add(y);
    }
    catch (ClassCastException e)
    {
        System.out.println("La clase a la que pertenece: " + obj.getClass().getName()
        );
        e.printStackTrace();
    } // fin de catch

```

InputMismatchException:

Especificación:

```
public class InputMismatchException extends NoSuchElementException
```

¿Cuándo Ocurre?:

Esta excepción es lanzada por un objeto `Scanner` para indicar que la entrada recibida por el usuario no coincide con el patrón del tipo esperado, o que la entrada (token) está por fuera del rango del tipo esperado.

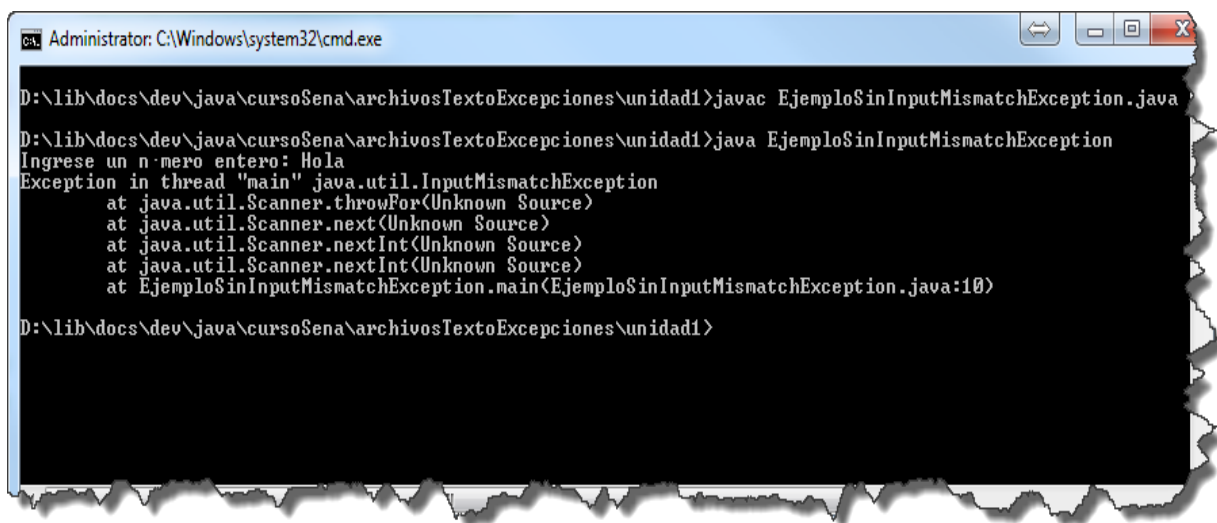
¿Cómo se evita?:

En este ejemplo, la excepción **InputMismatchException** es lanzada:

```
public class EjemploInputMismatchException
{
    public static void main( String args[] )
    {
        Scanner scanner = new Scanner( System.in );

        System.out.print( "Ingrese un número entero: " );
        int entero = scanner.nextInt();

    } // fin del método main
} // fin de la clase EjemploInputMismatchException
```



```
Administrator: C:\Windows\system32\cmd.exe
D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>javac EjemploSinInputMismatchException.java
D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>java EjemploSinInputMismatchException
Ingrese un número entero: Hola
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Unknown Source)
    at java.util.Scanner.next(Unknown Source)
    at java.util.Scanner.nextInt(Unknown Source)
    at java.util.Scanner.nextInt(Unknown Source)
    at EjemploSinInputMismatchException.main(EjemploSinInputMismatchException.java:10)
D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>
```

A continuación se incluye una aplicación que trata esta excepción y permite que el usuario pueda ingresar el número de nuevo:

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class EjemploInputMismatchException
{
    public static void main( String args[] )
    {
        Scanner scanner = new Scanner( System.in );
```

```

        boolean continuarCiclo = true; // determine if more input is
needed
        do
        {
            try
            {
                System.out.print( "Ingrese un número entero: " );
                int entero = scanner.nextInt();

                continuarCiclo = false; // entrada satisfactoria

            } // end try
            catch ( InputMismatchException inputMismatchException )
            {
                System.err.printf( "\nExcepción: %s", inputMismatchException );
                scanner.nextLine(); //
                System.out.println( "Usted debe ingresar un número.\n"
);
            } // end catch
        } while ( continuarCiclo ); // fin del ciclo do...while
    } // fin de main
} // fin de la clase EjemploInputMismatchException

```

```

Administrator: C:\Windows\system32\cmd.exe - java EjemploInputMismatchException
D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>javac EjemploInputMismatchException.java
D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>java EjemploInputMismatchException
Ingrese un número entero: Hola
Excepción: java.util.InputMismatchExceptionUsted debe ingresar un número.
Ingrese un número entero:

```

Enlace: <http://paste.ubuntu.com/704884/>

IllegalArgumentException:

Especificación:

public class **IllegalArgumentException** extends RuntimeException

¿Cuándo Ocurre?:

Una excepción Java que puede ser lanzada deliberadamente por métodos que no son compatibles con los tipos de los parámetros.

¿Cómo se evita?:

```
import java.lang.IllegalArgumentException;

public class EjemploIllegalArgumentException
{
    public static void main( String args[] )
    {

        double[] numeros = new double[ 0 ];

        System.out.printf( "Promedio: %.2f\n", promedio( numeros ) );

    } // fin de main

    public static double promedio( double [] arreglo ) throws
    IllegalArgumentException
    {
        if ( arreglo.length == 0 )
        {
            throw new IllegalArgumentException();
        } // fin de else
        else
        {
            double suma = 0.0;

            for ( int i = 0; i < arreglo.length; i++ )
                suma += arreglo[ i ];

            return suma / arreglo.length;
        } // fin de else
    } // fin del método promedio
} // fin de la clase EjemploInputMismatchException
```

FileNotFoundException:

Especificación:

public class **FileNotFoundException** extends IOException

¿Cuándo Ocurre?:

Esta excepción será lanzada por los constructores de las clases `FileInputStream`, `FileOutputStream`, y `RandomAccessFile` cuando un archivo especificado en la ruta no existe. También será lanzada por estos constructores si el archivo existe pero algún motivo es inaccesible, por ejemplo cuando se hace un intento de abrir un archivo de sólo escritura para ser escrito.

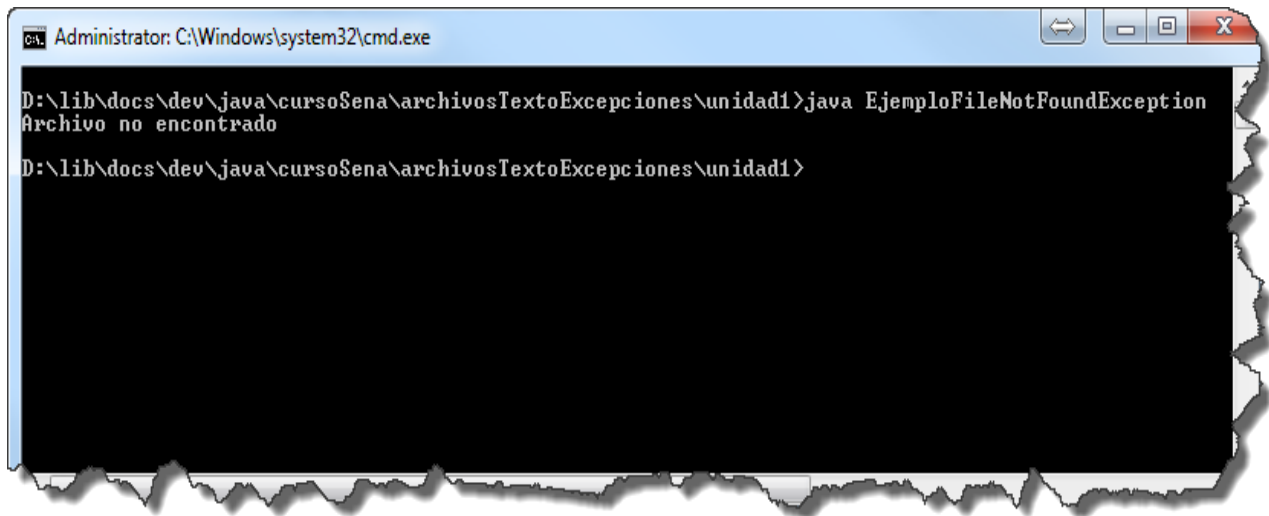
¿Cómo se evita?:

```
import java.io.*;

public class EjemploFileNotFoundException
{
    public static FileInputStream flujoEntrada(String nombreArchivo) throws
FileNotFoundException
    {
        FileInputStream fis = new FileInputStream(nombreArchivo);
        return fis;
    } // fin del método

    public static void main(String args[])
    {
        FileInputStream fis = null;
        String nombreArchivo = "C:\\datos.txt";

        // get file input stream 1
        try
        {
            fis = flujoEntrada( nombreArchivo );
        }
        catch ( FileNotFoundException ex )
        {
            System.out.println("Archivo no encontrado");
        } // fin de catch
    } // fin de main
} // fin de la clase
```

Enlace: <http://paste.ubuntu.com/705150/>

Excepciones encadenas:

Ejemplo práctico de cómo tratar excepciones encadenadas. Este tipo de excepciones es el resultado de un serie de *lanzamientos* de irregularidades en métodos. Cada método se encarga de atrapar la excepción producida por el método que desde su implementación se invoca. Finalmente, la pila de excepciones se encarga de mostrar un volcado de la cadena.

```
public class UsandoExcepcionesEncadenadas
{
    public static void main( String args[] )
    {
        try
        {
            metodo1();
        } // end try
        catch (Exception exception)
        {
            exception.printStackTrace();
        } // end catch

    } // main

    // invoca a metodo2;
    private static void metodo1() throws Exception
    {
        try
        {
            metodo2(); // call metodo2
        } // fin de try
        catch (Exception exception)
        {
            // ...
        }
    }
}
```

```

        throw new Exception( "Excepción lanzada en metodo1", exception );
    } // fin de catch
} // fin de metodo1

// invoca a metodo3
private static void metodo2() throws Exception
{
    try
    {
        metodo3(); //
    } // fin de try
    catch ( Exception exception )
    {
        throw new Exception( "Excepción lanzada en metodo2", exception );
    } // fin de catch
} // fin de metodo2

// Excepción lanzada de retorno al método metodo2
private static void metodo3() throws Exception
{
    throw new Exception( "Excepción lanzada en metodo3" );
} // fin de metodo3
} // fin de la clase UsandoExcepcionesEncadenadas

```

```

Administrator: C:\Windows\system32\cmd.exe

D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>javac UsandoExcepcionesEncadenadas.java
D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>java UsandoExcepcionesEncadenadas
java.lang.Exception: Excepción lanzada en metodo1
    at UsandoExcepcionesEncadenadas.metodo1(UsandoExcepcionesEncadenadas.java:25)
    at UsandoExcepcionesEncadenadas.main(UsandoExcepcionesEncadenadas.java:7)
Caused by: java.lang.Exception: Excepción lanzada en metodo2
    at UsandoExcepcionesEncadenadas.metodo2(UsandoExcepcionesEncadenadas.java:38)
    at UsandoExcepcionesEncadenadas.metodo1(UsandoExcepcionesEncadenadas.java:21)
    ... 1 more
Caused by: java.lang.Exception: Excepción lanzada en metodo3
    at UsandoExcepcionesEncadenadas.metodo3(UsandoExcepcionesEncadenadas.java:45)
    at UsandoExcepcionesEncadenadas.metodo2(UsandoExcepcionesEncadenadas.java:34)
    ... 2 more

D:\lib\docs\dev\java\cursoSena\archivosTextoExcepciones\unidad1>

```

Enlace: <http://paste.ubuntu.com/705201/>